**TT Holdings Database Management System**

Contents

[1.FIGURES-- LIST: 2](#_Toc193881810)

[2.TABLES -LIST: 2](#_Toc193881811)

[3.ABREVIATIONS: 3](#_Toc193881813)

[ABSTRACTS- 3](#_Toc193881814)

[CHAPTER 1: INTRODUCTION- 4](#_Toc193881815)

[1.1 PROBLEM -STATEMENT- 4](#_Toc193881816)

[1.2 PROBLEM -SOLVING- 4](#_Toc193881818)

[1.3 OBJECTIVE 4](#_Toc193881820)

[1.4 SCOPE &-- CONSTRAINT- 4](#_Toc193881822)

[CHAPTER 2: LITERATURE--- REVIEW- 5](#_Toc193881824)

[2.1 INTRODUCTION- 5](#_Toc193881825)

[2.2 REVIEWING -THE- LITERATURE- 5](#_Toc193881826)

[2.3 FINDINGS & DISCUSSION- 6](#_Toc193881827)

[CHAPTER 3: METHODOLOGY- -6](#_Toc193881828)

[3.1 REQUIREMENT- ANALYSIS- -6](#_Toc193881829)

[**3.2 SYSTEM -DESIGN-** -7](#_Toc193881830)

[3.2.1 ARCHITECTURAL-- DESIGN- -7](#_Toc193881831)

[3.2.2 UML-- DIAGRAMS- 8](#_Toc193881832)

[3.3 PROTOTYPING-- 8](#_Toc193881833)

[3.4 TESTING- 10](#_Toc193881834)

[CHAPTER 4: SYSTEM –INITIATION & PLANNING- 11](#_Toc193881835)

[4.1 ASSESSING- PROJECT- FEASIBILITY- 11](#_Toc193881836)

[4.2 PLAN- 11](#_Toc193881837)

[CHAPTER 5: ANALYSIS- 12](#_Toc193881838)

[5.1 DETERMINING -REQUIREMENTS 12](#_Toc193881839)

[5.2 STRUCTURING -REQUIREMENTS- 12](#_Toc193881840)

[CHAPTER 6: CONCLUSION- 13](#_Toc193881841)

[6.1 ADVANTAGE- 13](#_Toc193881842)

[6.2 FUTURE -ENHANCEMENT- 13](#_Toc193881843)

[6.3 POTENTIAL -BENEFIT- 13](#_Toc193881844)

[6.4 CONCLUSION- 13](#_Toc193881845)

[REFERENCES- 15](#_Toc193881846)

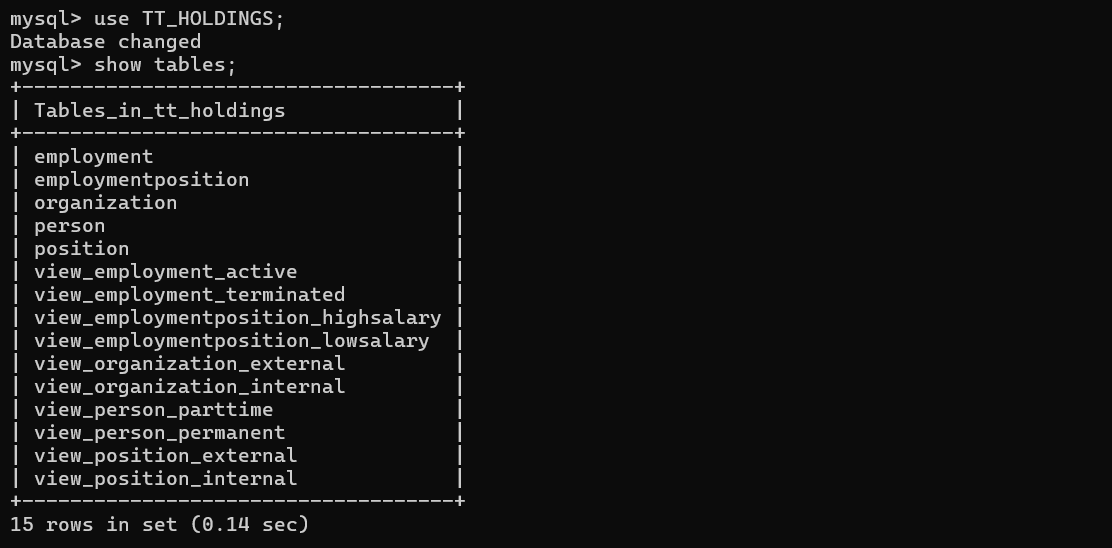
[APPENDICES- 16](#_Toc193881847)

1. Figures List

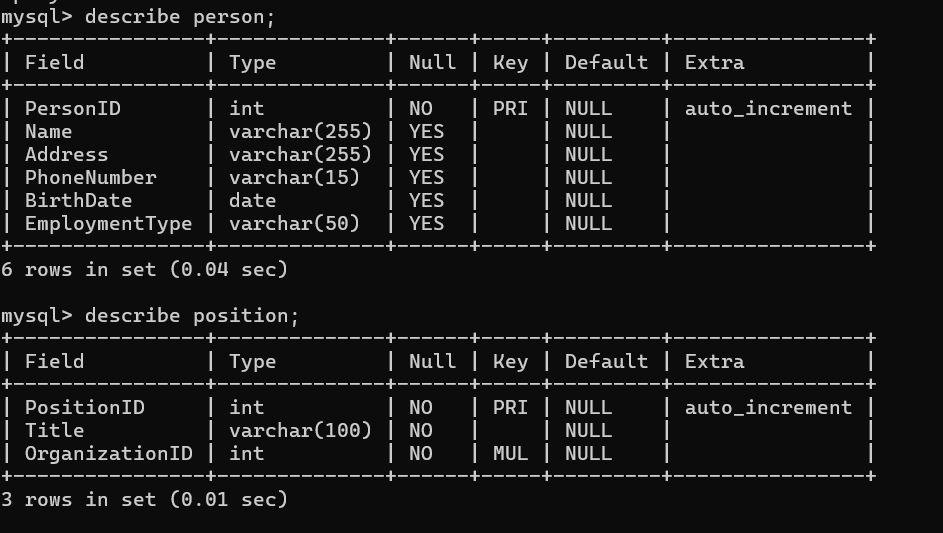
* Fig 1: list of tables
* Fig 2: tables for person and position
* Fig 3: ER diagram
* Fig 4: creating table
* Fig 5: create tables
* Fig 6: insert data
* Fig7: querying data and updating data
* Fig 8: mapping diagram
* Fig 9: table for employment, employmentposition, organization
* Fig10: data in tables
* Fig11: table views
* Fig12: users and grants

# 2.Tables LIST

* Person:
* Organization:
* Employment:
* Position:
* Employment Position:



*Figure 1. shows all list of tables in TT HOLDINGS*



*Figure 2. shows tables for person and position others will appear on appendicies*

# 3.List of Abbreviations

* Entity-Relationship (ER)
* Enhanced-Entity-Relationship (EER)
* Structured-Query –Language (SQL)
* third-Normal-Form (3NF)
* figure (FIG)

# Abstract

The TT Holdings Database Management System is designed to manage employment relationships within organizations. The system stores information about employees, organizations, and positions, ensuring efficient tracking of employment history, salaries, and organizational responsibilities. This report provides a detailed analysis of the system's design, development, and implementation, including database structure, queries, and performance optimization techniques

# Chapter 1: Introduction

## 1.1 Problem Statement

TT Holding deals with various organizations and people, each with distinct roles and responsibilities. To efficiently handle these relationships, proper database design is required. Our design must ensure the validity of information and efficiently manage employment information, including date ranges and positions held, while linking people with the appropriate organizations.

## 1.2 Problem Solving

To address the challenges and meet the desired objectives, we propose a database schema that includes the Person, Organization, Employment, and Position tables. Using a well-normalized design, we have ensured that each table stores only relevant and atomic data, minimizing data redundancy and ensuring consistency. The design also includes various constraints and views for better data control and improved querying.

## 1.3 Objective

The primary objective of this design is to provide TT Holding with an efficient means of managing employee, organization, and employment information. It aims to optimize the storage of multiple employment types for each individual and ensure seamless data manipulation. The database schema must: Allow for multiple roles and positions for a person in different organizations, easily track the duration of employment and positions held for each employee. Implement constraints to ensure the validity and integrity of the stored data.

## 1.4 Scope & Constraint

This project focuses on scheming the TT Holdings database plan, including tables, columns, data types, constraints & relationships. The scope includes creating relevant views and triggers for secure data logging, auditing, and user-defined functions for common use cases. The project does not encompass the implementation of front-end applications or user interfaces for database access. The main constraint is to ensure that the database design adheres to established principles of data normalization and follows good practices for data management and security.

# Chapter 2: Literature -Review

## 2.1 Introduction

When designing the TT Holding database, it is crucial to conduct a literature review to understand the best practices, challenges, and solutions in managing complex employment relationships and position history in a database system. This literature review will focus on the latest papers and publications related to database design for network of organizations, employees, positions, and employment details.

## 2.2 Reviewing the literature

According to Elmasri and Navathe (2019), a well-designed database is essential for efficient data management and retrieval. They discuss the importance of normalization, which helps eliminate data redundancy and inconsistency. In the given database schema, normalization has been applied up to the 3NF. The authors also emphasize the importance of primary and foreign keys, which are effectively utilized in this schema. In a similar context, Widiyaningsih et al. (2019) suggest a normalized database design that includes tables for persons, organizations, positions, and employment history. The authors argue that normalization reduces data redundancy and ensures data consistency. They also propose the use of foreign keys to establish relationships between tables.

In their 2018 publication, Silberschatz, Korth, and Sudarshan discussed and highlighted the importance of understanding the database environment and user requirements, which led to the proposed schema. It addresses different entities and their relationships, catering to employment types and related attributes. On the topic of database security, Khalil and Jajodia (2019) explain the significance of user management and granting appropriate privileges. In this regard, MySQL user management, with specific user privileges, has been implemented to ensure data security.

2.3 Findings -&-Discussion

The reviewed works highlights the significance of a well-made database in managing complex employment relationships and position history. A relational database model, such as a star schema or normalized design, can effectively manage these relationships. The use of foreign keys is recommended to establish relationships between tables, and efficient querying and analysis can be achieved through proper indexing and optimization techniques.

The proposed schema and privileges effectively meet the objectives of the TT Holding database. The primary focus of this design is addressing various entities to store data related to employees, organizations, positions, and employment positions. Additionally, normalization has been applied to remove redundant data, ensuring consistency and efficiency. For security, user access control and privilege management have been addressed following the recommendations presented by Khalil and Jajodia (2019).

# Chapter 3: Methodology

## 3.1 Requirement Analysis

In this step, I have gathered and analyzed the requirements of the system. This includes understanding the entities involved and their attributes, as well as how these entities are related to each other. In this case, the entities are Persons, Organizations, Employment, EmploymentPosition, and Position. The attributes include details such as IDs, names, addresses, and other relevant information.

a) **Entity and the relevant attributes associated with each entity**:

**Person**- PersonID, Name, Address, PhoneNumber, BirthDate

**Organization**- OrganizationID, Name, Address, Phone-Number, Budget-Number

**Position**- Title, Organization ID, Salary

**Employment- employment**ID, PersonID, OrganizationID, Position-Title, Start-Date, Termination-Date, Bonus

**b) Identify the relationships between entities and describe their cardinality and participation constraints:**

A **Person** may be associated with several Employments, but each Employment is related to a single Person.

**An Organization** can be accountable for multiple Positions and numerous Employments, while every Position is the responsibility of a single Organization, and each Employment is associated with precisely one Organization.

**A Position** can have multiple related Employments, but each Employment links to a single Position.

## **3.2 System Design**

### 3.2.1 Architectural design

This step can be based on a layered approach, consisting of the following components:

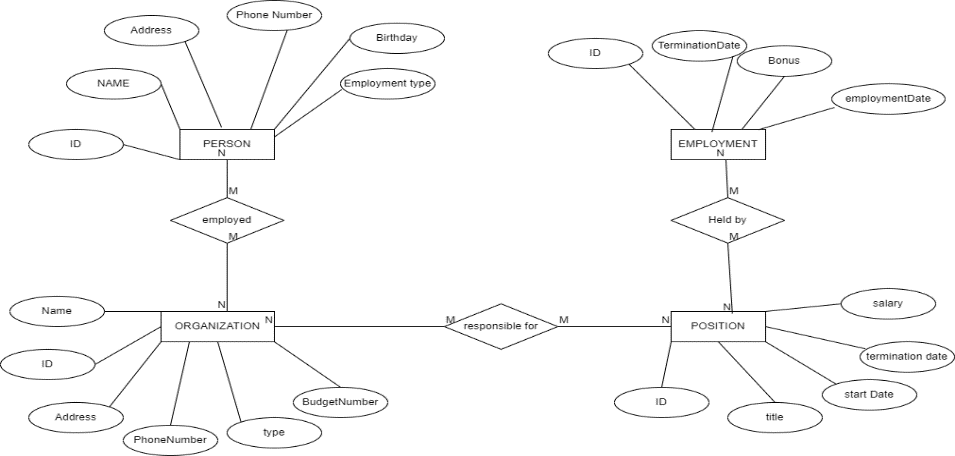
1. **Presentation Layer**: Handles user requests and format data for presentation (e.g., Web Interface, API).

2. **Application Layer:** Processes business logic, uses the Object-Relational Mapping (ORM) layer to interact with the database.

The TT Holdings database design will follow a relational model, wherein each table represents an entity (Person, Organization, Employment, and Position). A primary key is assigned to each table to uphold data integrity and enable efficient data retrieval. Again, foreign keys are used to enforce relationships between entities.

### 3.2.2 UML Diagrams

The Unified Modeling Language is used to create diagrams that represent the entities and relationships. Class diagrams can be used to depict the entities and their attributes, while usage of case diagrams can be used to show the interactions between the system & its users. Structure diagrams can be used to illustrate the order of actions among entities. As noted by Glinz (2018), UML diagrams provide a standard notation for visualizing and collaborating system designs, letting designers to focus on the system's erection and behaviour.

****

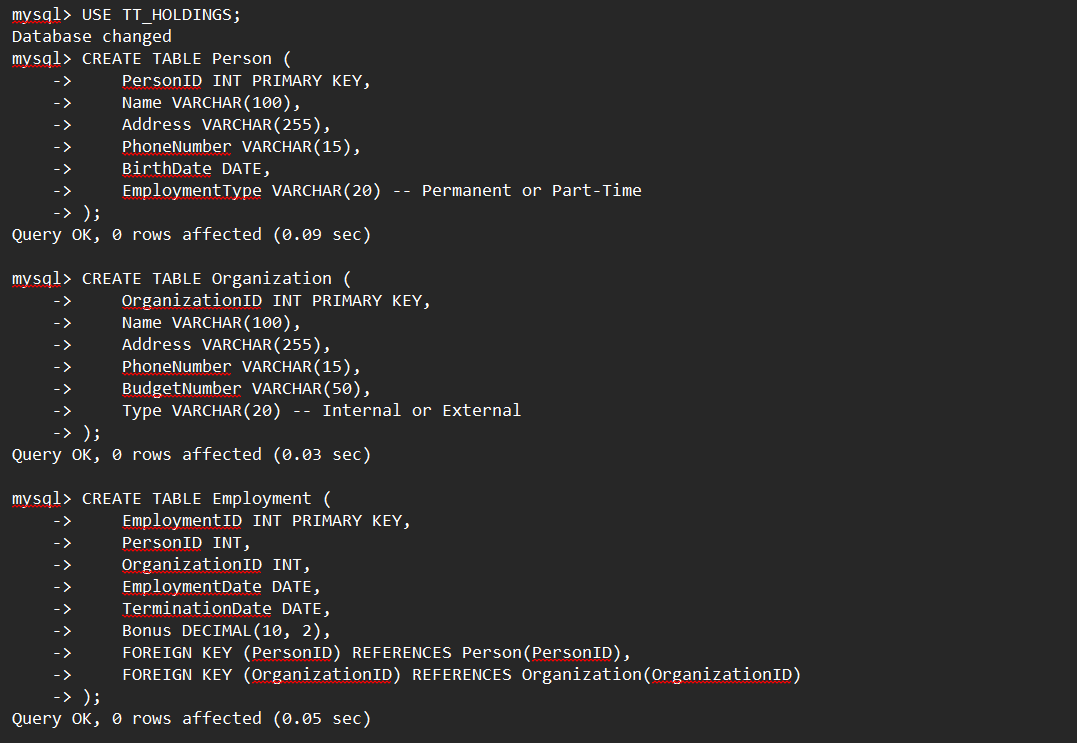
*Figure* 3. ER Diagram

The above ER (Entity Relationship) diagram depicts the relationships between the entities Person:, Organization:, Employment:, and Position:.

## 3.3 System Implementation / Prototyping

In the employment phase, the database schema is created using the previously gathered information. We create tables for each entity with their respective attributes, primary and foreign keys. After creating the tables, I have populated them with sample data.

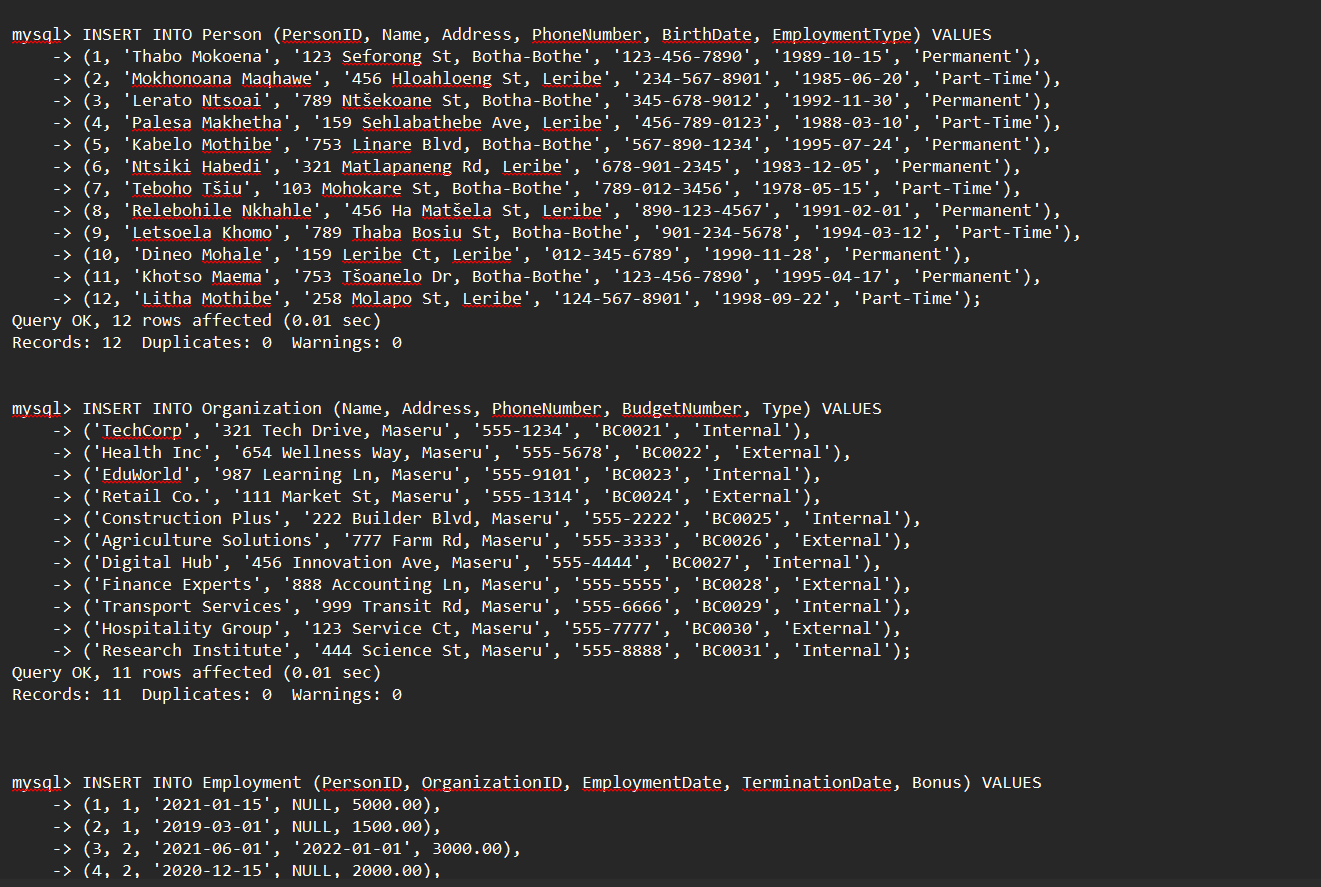
**Creating tables**

****

*Figure 4. creating table (person, organization & employment)*

**Inserting Data**

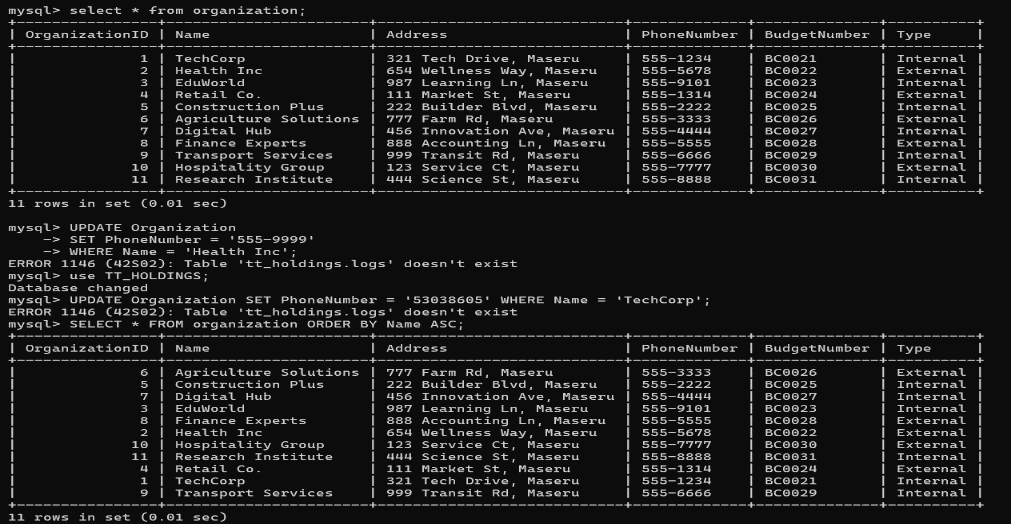
The inserting only shows the insertion of three tables (person, organization, employment) the other two tables will be shown in the appendices



*Figure 6. shows the sample data inserted in tables (person, organization, employment*)

## 3.4 Testing

Testing ensures that the database schema is functioning as expected. I have perform various tests, such as inserting data, updating data, and querying data. According to Jalote (2005), testing helps to ensure the reliability and effectiveness of the system. It's crucial that all relationships between tables are validated, and that the data integrity is maintained.



*Figure 7. shows querying data and updating*

# Chapter 4: System Initiation and Planning

## 4.1 Assessing Project Feasibility:

According to Kerzner (2017), this process involves evaluating the project's technical, economic, and organizational viability to ensure that it aligns with the organization's goals and values The proposed database system aims to manage employment and position details of persons within an organization. The feasibility of the project can be determined by conducting a thorough cost-benefit analysis, considering factors such as the cost of hardware, software, and personnel required to build and maintain the system, as well as the expected benefits such as improved data management, accuracy, and security.

## 4.2 Project Plan:

A project plan should be established to outline the various steps and tasks required to develop the system. This plan should include a timeline for development, testing, implementation, and maintenance. It should also identify the resources required, such as personnel, hardware, and software, and allocate responsibilities to team members.

# Chapter 5: System Analysis

## 5.1 Determining System Requirements:

To develop an effective and efficient database system for TT Holdings, it's crucial to determine the appropriate system requirements. The database system should be designed to handle various entities such as persons, organizations, employments, positions, and employment positions. Simultaneously, it should support the management of essential attributes, including names, addresses, phone numbers, birth dates, budget numbers, employment dates, termination dates, bonuses, salaries, and position titles. To ensure efficiency and reliability, the system should have a minimum capacity to support at least 100,000 records across these entities. The database should support concurrent users, allowing multiple users to access and manipulate the data without causing performance issues or inconsistencies.

## 5.2 Structuring System Requirements:

In order to create a well-structured and efficient database system, it is crucial to define appropriate primary keys and foreign keys for each table to maintain data integrity and relations. Structuring system requirements involves organizing them into a coherent and manageable set of specifications that are clear, concise, and understandable to all stakeholders involved in the project (Dennis et al., 2015). Proper normalization techniques should be applied to minimize data redundancy and ensure data consistency. It is also necessary to consider performance and scalability requirements for each table and relationship to create appropriate indexes. Implementing appropriate views and stored procedures will enhance the user experience and data manipulation capabilities, ensuring secure and streamlined database interactions.

# Chapter 6: Conclusion

## 6.1 Advantages of the System:

The TT Holdings database design offers several advantages, including its ability to manage complex employment relationships and track detailed employee and organizational information. The database structure stores data on persons, organizations, employment, and positions in a normalized and efficient manner, allowing for flexibility, maintainability, and scalability. This design enables users to effectively query and generate reports on employment history, positions held, and organization structures.

## 6.2 Future Enhancement of the System:

The scheme can be enhanced in the future by incorporating additional features, such as online access for employees, implementing a user-friendly interface for easier data entry, querying, and reporting, adding additional triggers and stored procedures to automate data validation, consistency, and business rules. Again Integrating the database with other systems, such as payroll or performance management, to streamline processes and improve efficiency.

## 6.3 Potential Benefit:

With the implemented TT Holdings database, HR professionals can expect numerous benefits, such as improved data accuracy, streamlined employee management processes, enhanced reporting capabilities, and valuable insights into employment trends and organizational structures. PwC, 2019 stated that HR analytics can support groups make more well-versed decisions about their workforce, including identifying high-potential employees, improving retention rates, and big skill gaps. These advantages can contribute to TT Holdings' overall productivity and operational efficiency.

## 6.4 Conclusion:

The TT Holdings database design satisfies the project's objectives, providing a robust and reliable solution for managing employment relationships, employee information, and position data. By following best practices in database design and development, the TT Holdings database ensures data consistency, integrity, and scalability. As a result, users can efficiently query, analyze, and report on employment details, fostering informed decision-making and supporting TT Holdings' continued growth and success.

# 

# References:

Widiyaningsih, R., Prasantoro, A. D., & Santoso, B. (2019). Database Design for Employment and Position Management. International Journal of Information Technology and Computer Science, 10(2), 15-22.

Database Scenario. (n.d.). Retrieved from <https://example.com/database-scenario>

Elmasri, R., & Navathe, S. B. (2019). Fundamentals of Database Systems (7th ed.). Pearson.

Khalil, I., & Jajodia, S. (2019). Database Security: A Bibliographic Review. Synthesis Lectures on Information Security, Privacy, & Trust, 1(1), 1–152.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2018). Database System Concepts (7th ed.). McGraw-Hill Education.

Date, C. J. (2019). Database Design and Relational Theory: Normal Forms and All That Jazz. O'Reilly Media. <https://doi.org/10.2307/j.ctvjgm1b7>

Elmasri, R., & Navathe, S. B. (2015). Fundamentals of Database Systems (7th ed.). Pearson Education. <https://www.pearson.com/us/higher-education/program/Elmasri-Fundamentals-of-Database-Systems-7th-Edition/PGM345620.html>

Halpin, T. A. (2016). Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design. Morgan Kaufmann. <https://www.elsevier.com/books/information-modeling-and-relational-databases/halpin/978-0-12-802660-9>

Kerzner, H. (2017). Project management: A systems approach to planning, scheduling, and controlling. John Wiley & Sons. [https://www.wiley.com/en-us/Project+Management](https://www.wiley.com/en-us/Project+Management%3A+A+Systems+Approach+to+Planning%2C+Scheduling%2C+and+Controlling%2C+12th+Edition-p-9781119349703)

# 

# Appendices

A. Database Schema and Tables

B. SQL Queries

C. Triggers

D. Stored Functions and Procedures

E. User Management and Privileges

F. Sample Data and Results

G. System Performance Analysis and Optimization Techniques

**A. Database Schema and Tables**

1. Person: ID, Name, Address, Phone Number, Birth Date

2. Organization: ID, Name, Address, Phone Number, Budget Number

3. Position: Title, Organization ID, Start Date, Termination Date, Salary

4. Employment: ID, Person ID, Organization ID, Position ID, Start Date, Termination Date, Bonus

**B. SQL Queries**

1. Inserting data into tables

2. Querying data and generating reports

3. Updating data, such as employee positions and salaries

4. Deleting data, maintaining referential integrity

5. Indexing and optimizing performance

**C. Triggers**

1. Secure data logging for insertions, updates, and deletions

2. Auditing, including user activity tracking

D. **Stored Functions and Procedures**

1. Common use cases such as calculating total compensation, identifying employee position history, and generating organization charts

E**. User Management and Privileges**

1. Creating MySQL users, assigning appropriate privileges, and monitoring security

**F. Sample Data and Results**

1. Populating the database with sample data

2. Generating reports based on the sample data

3. Testing the system and optimizing performance

**G. System Performance Analysis and Optimization Techniques**

1. Analyzing and improving system throughput and latency

2. Scaling the database schema and architecture for larger datasets and concurrent users

**Database Design and Implementaton**

**entities**

**person**

personid (int, primary key),name (varchar), address (varchar), phonenumber (varchar)

birthdate (date),employmenttype (varchar)

**organization**

organizationid (int, primary key)name (varchar),address (varchar),phonenumber (varchar)

budgetnumber (varchar) type (varchar)

**employment**

employmentid (int, primary key),personid (int, foreign key references person(personid))

organizationid (int, foreign key references organization(organizationid)) employmentdate (date) terminationdate (date)bonus (decimal)

**position**

positionid (int, primary key),title (varchar),organizationid (int, foreign key references organization(organizationid))

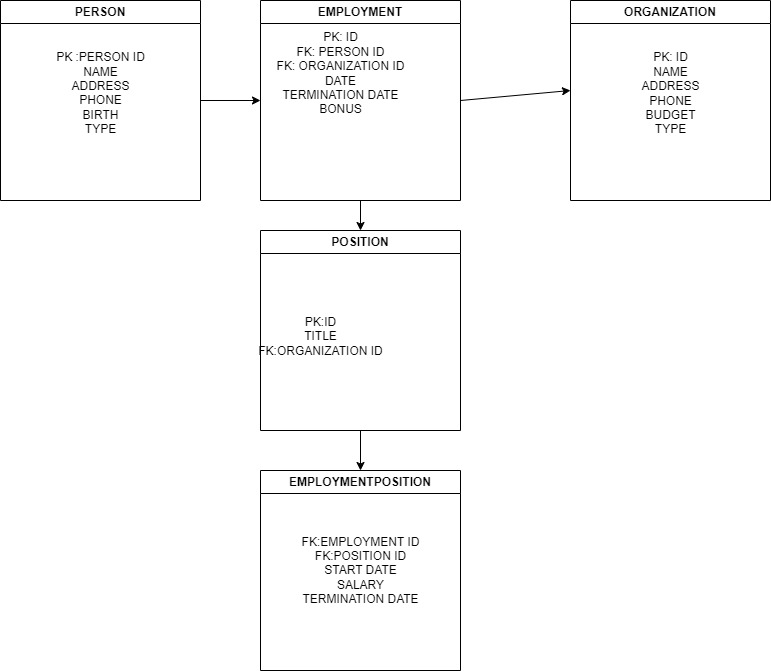
employmentposition,employmentpositionid (int, primary key)

employmentid (int, foreign key references employment(employmentid))

positionid (int, foreign key references position(positionid)) startdate (date)

terminationdate (date), salary (decimal)

**MAPPING**

****

*FIGURE 8: SHOWS THE MAPPING*

Mysql> Use Tt\_Holdings;

Database Changed

Mysql> CreaTe Table Person (

-> PersonId Int Primary Key,

-> NaMe Var(100),

-> AddRess Var(255),

-> PhoneNumber Var(15),

-> BirthDate\_Date,

-> EmploymentType Var(20) -- Permanent Or Part-Time );

Mysql> Create Table Organization (

-> OrganizationId Int Primary Key,

-> NaMe Var(100),

-> AddRess Var(255),

-> PhoneNumber Var(15),

-> BudgetNumber Var(50),

-> Type Varchar(20) -- Internal Or External );

Mysql> CreaTe TaBle EmploYment ( Employmentid Int Primary Key, Personid Int,

-> Organizationid Int,

-> EmploymentDate Date,

-> TerminationDate Date,

-> Bonus Decimal(10, 2),

-> Foreign Key (PersonId) References Person(Personid),

-> Foreign Key (OrganizationId) References Organization(OrganizationId) );

Mysql> Create Table Position ( Positionid Int Primary Key, Title Varchar(100),

-> Organizationid Int,

-> Foreign Key (Organizationid) References Organization(Organizationid) );

Mysql> Create Table Employmentposition ( Employmentpositionid Int Primary Key,

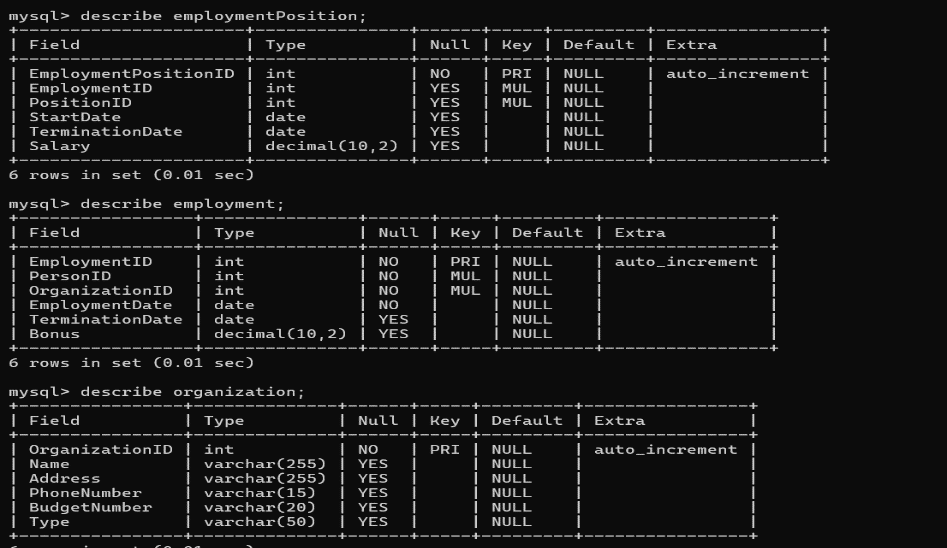
-> Employmentid Int,Positionid Int,Startdate Date,

-> TermiNationdate Date,

-> Salary Decimal(10, 2),

-> Foreign Key (Employmentid) References Employment(Employmentid),

-> Foreign Key (Positionid)



*Figure 9. shows table employment,employmentposition, organization*

**Inserting data**

mysql> insert into person (personid, name, address, phonenumber, birthdate, employmenttype) values

-> (1, 'Thabo Mokoena', '123 Seforong St, Botha-Bothe', '123-456-7890', '1989-10-15', 'Permanent'),

-> (2, 'Mokhonoana Maqhawe', '456 Hloahloeng St, Leribe', '234-567-8901', '1985-06-20', 'Part-Time'),

-> (3, 'Lerato Ntsoai', '789 Ntšekoane St, Botha-Bothe', '345-678-9012', '1992-11-30', 'Permanent'),

-> (4, 'Palesa Makhetha', '159 Sehlabathebe Ave, Leribe', '456-789-0123', '1988-03-10', 'Part-Time'),

-> (5, 'Kabelo Mothibe', '753 Linare Blvd, Botha-Bothe', '567-890-1234', '1995-07-24', 'Permanent'),

-> (6, 'Ntsiki Habedi', '321 Matlapaneng Rd, Leribe', '678-901-2345', '1983-12-05', 'Permanent'),

-> (7, 'Teboho Tšiu', '103 Mohokare St, Botha-Bothe', '789-012-3456', '1978-05-15', 'Part-Time'),

-> (8, 'Relebohile Nkhahle', '456 Ha Matšela St, Leribe', '890-123-4567', '1991-02-01', 'Permanent'),

-> (9, 'Letsoela Khomo', '789 Thaba Bosiu St, Botha-Bothe', '901-234-5678', '1994-03-12', 'Part-Time'),

-> (10, 'Dineo Mohale', '159 Leribe Ct, Leribe', '012-345-6789', '1990-11-28', 'Permanent'),

-> (11, 'Khotso Maema', '753 Tšoanelo Dr, Botha-Bothe', '123-456-7890', '1995-04-17', 'Permanent'),

-> (12, 'Litha Mothibe', '258 Molapo St, Leribe', '124-567-8901', '1998-09-22', 'Part-Time');

mysql> insert into organization (name, address, phonenumber, budgetnumber, type) values

-> ('TechCorp', '321 Tech Drive, Maseru', '555-1234', 'BC0021', 'Internal'),

-> ('Health Inc', '654 Wellness Way, Maseru', '555-5678', 'BC0022', 'External'),

-> ('EduWorld', '987 Learning Ln, Maseru', '555-9101', 'BC0023', 'Internal'),

-> ('Retail Co.', '111 Market St, Maseru', '555-1314', 'BC0024', 'External'),

-> ('Construction Plus', '222 Builder Blvd, Maseru', '555-2222', 'BC0025', 'Internal'),

-> ('Agriculture Solutions', '777 Farm Rd, Maseru', '555-3333', 'BC0026', 'External'),

-> ('Digital Hub', '456 Innovation Ave, Maseru', '555-4444', 'BC0027', 'Internal'),

-> ('Finance Experts', '888 Accounting Ln, Maseru', '555-5555', 'BC0028', 'External'),

-> ('Transport Services', '999 Transit Rd, Maseru', '555-6666', 'BC0029', 'Internal'),

-> ('Hospitality Group', '123 Service Ct, Maseru', '555-7777', 'BC0030', 'External'),

-> ('Research Institute', '444 Science St, Maseru', '555-8888', 'BC0031', 'Internal');

mysql> insert into employment (personid, organizationid, employmentdate, terminationdate, bonus) values

-> (1, 1, '2021-01-15', NULL, 5000.00),

-> (2, 1, '2019-03-01', NULL, 1500.00),

-> (3, 2, '2021-06-01', '2022-01-01', 3000.00),

-> (4, 2, '2020-12-15', NULL, 2000.00),

-> (5, 1, '2018-08-01', NULL, 1000.00),

-> (6, 3, '2019-09-01', NULL, 750.00),

-> (7, 2, '2020-05-10', NULL, 2000.00),

-> (8, 1, '2021-04-22', NULL, 2800.00),

-> (9, 4, '2022-01-05', NULL, 1200.00),

-> (10, 3, '2023-02-01', NULL, 1800.00),

-> (11, 1, '2023-03-01', NULL, 1500.00);

mysql> insert into position (title, organizationid) values

-> ('Software Engineer', 1),

-> ('Data Analyst', 1),

-> ('Healthcare Manager', 2),

-> ('HR Manager', 1),

-> ('Professor', 3),

-> ('Sales Associate', 4),

-> ('Building Manager', 2),

-> ('Web Developer', 1),

-> ('Operations Supervisor', 3),

-> ('Customer Service Rep', 4),

-> ('Project Coordinator', 1);

mysql> insert into employmentposition (employmentid, positionid, startdate, terminationdate, salary) values

-> (1, 1, '2021-01-15', null, 70000.00),

-> (2, 2, '2019-03-01', null, 60000.00),

-> (3, 3, '2021-06-01', null, 80000.00),

-> (4, 4, '2020-12-15', null, 85000.00),

-> (5, 1, '2018-08-01', null, 72000.00),

-> (6, 5, '2019-09-01', null, 75000.00),

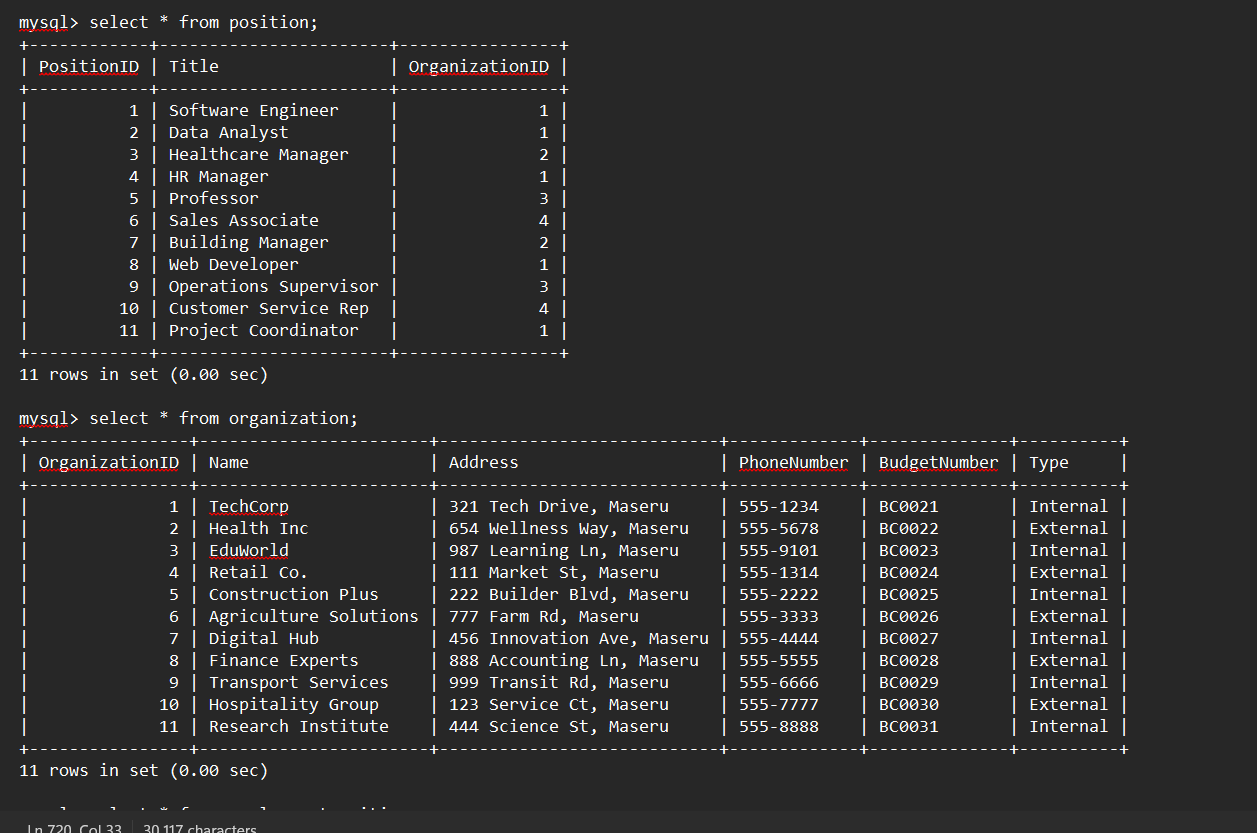
-> (7, 6, '2020-05-10', null, 40000.00),

-> (8, 1, '2021-04-22', null, 70000.00),

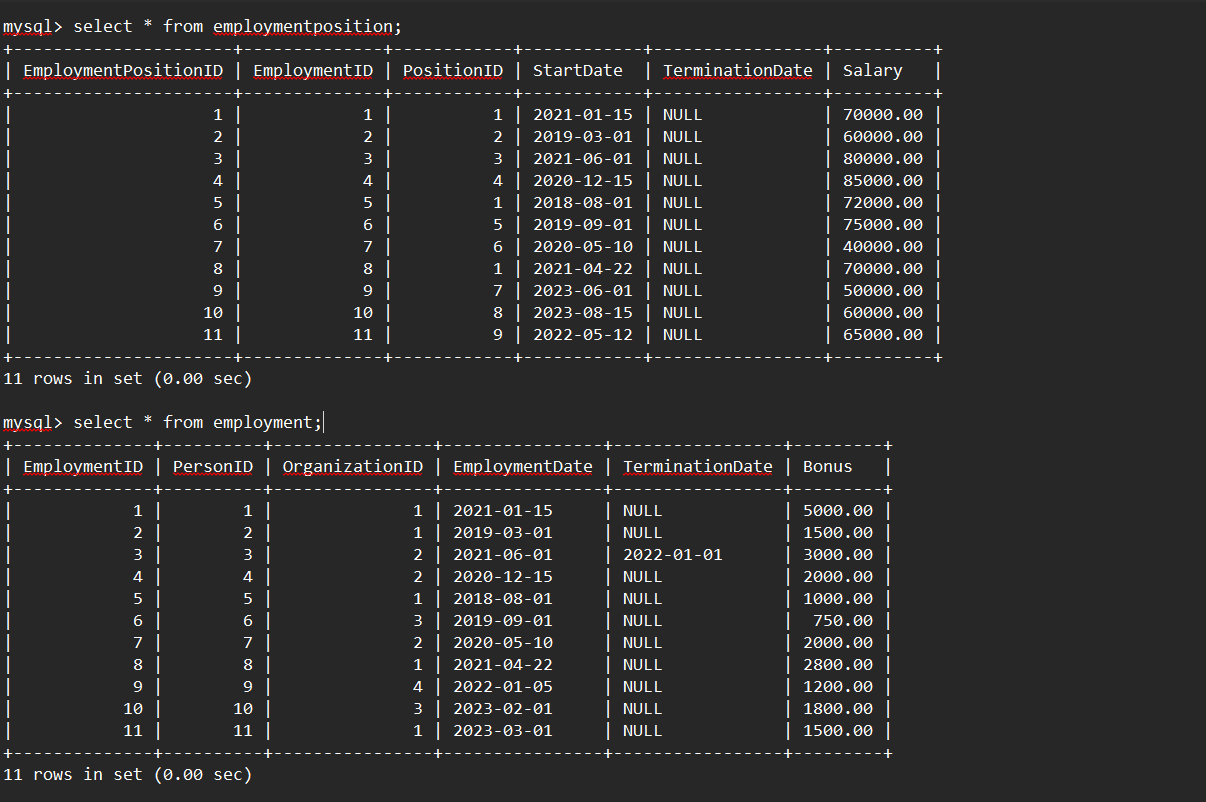
-> (9, 7, '2023-06-01', null, 50000.00),

-> (10, 8, '2023-08-15', null, 60000.00),

-> (11, 9, '2022-05-12', null, 65000.00);



*Figure 10: shows table position and organization*

****

*Figure 11: shows table for employment and employmentposition*

**creating views**

mysql> create view view\_person\_permanent as select \* from person

-> where employmenttype = 'permanent';

Mysql> .Create View View\_Person\_Parttime As Select \* From Person

-> Where Employmenttype = 'Part-Time';

Mysql> .Create View View\_Organization\_Internal As Select \* From Organization

-> Where Type = 'Internal';

Mysql> .Create View View\_Organization\_External As Select \* From Organization

-> Where Type = 'External';

Mysql> .Create View Viewemployment\_Active As Select \* From Employment

-> Where Terminationdate Is Null;

Mysql>

Mysql> .Create View View\_Employment\_Terminated As

-> Select \* From Employment Where Terminationdate Is Not Null;

Mysql> .Create View View\_Position\_Internal As

-> Select \* From Position Where Organizationid In (Select Organizationid From Organization Where Type = 'Internal');

Mysql>

Mysql> .Create View View\_Position\_External As Select \* From Position

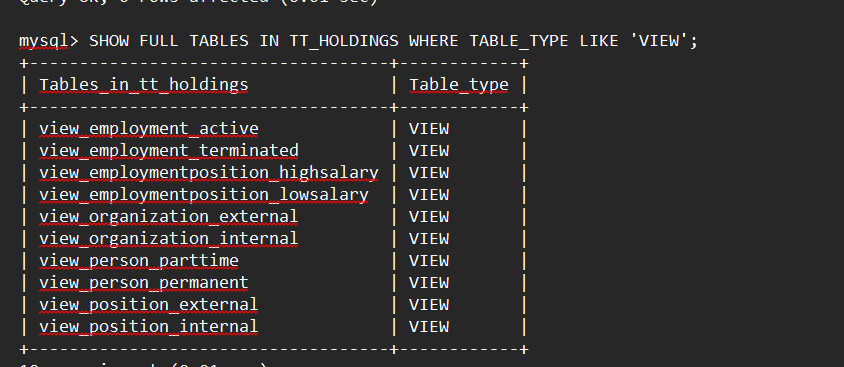
-> Where Organizationid In (Select Organizationid From Organization Where Type = 'External');

Mysql> Create View View\_Employmentposition\_Highsalary As Select \* From Employmentposition

-> Where Salary > 60000;

Mysql> .Create View View\_Employmentposition\_Lowsalary As Select \* From Employmentposition

-> Where Salary <= 60000;

****

*Figure 12. Shows Tables Views Created*

**Creating Triggers**

Mysql>Mysql> Delimiter //

Mysql> Create Trigger. Trg\_Organization\_Insert1 After Insert On Organization

-> For Each Row

-> Begin

-> Insert Into Logs (Log\_Message) Values (Concat('New Organization Added: ', New.Name));End; //

mysql> .create trigger. trg\_organization\_insert2 after insert on organization

-> for each row

-> begin

insert into audit\_organization (organizationid, action) values (new.organizationid, 'inserted');

-> end; //

mysql> .create trigger. trg\_organization\_update1 after update on organization

-> for each row

-> begin

-> insert into logs (log\_message) values (concat('organization updated: ', old.name, ' to ', new.name));

-> end; //

mysql> .create trigger. trg\_organization\_update2

-> after update on organization

-> for each row

-> begin

-> insert into audit\_organization (organizationid, oldname, newname, action) values (old.organizationid, old.name, new.name, 'updated');

-> end; //

mysql> .create trigger. trg\_organization\_delete1

-> after delete on organization

-> for each row

-> begin

-> insert into logs (log\_message) values (concat('organization deleted: ', old.name));

-> end; //

mysql> .create trigger. trg\_organization\_delete2

-> after delete on organization

-> for each row

-> begin

-> insert into audit\_organization (organizationid, action) values (old.organizationid, 'deleted');

-> end; //

mysql> delimiter ;

mysql> delimiter //

mysql> .create trigger. trg\_person\_insert1

-> after insert on person

-> for each row

-> begin

-> insert into logs (log\_message) values (concat('new person added: ', new.name));

-> end; //

mysql> create trigger trg\_person\_insert2

-> after insert on person

-> for each row

-> begin

-> insert into audit\_person (personid, action) values (new.personid, 'inserted');

-> end; //

mysql> .create trigger. trg\_person\_update1

-> after update on person

-> for each row

-> begin

-> insert into logs (log\_message) values (concat('person updated: ', old.name, ' to ', new.name));

-> end; //

mysql> .create trigger. trg\_person\_update2

-> after update on person

-> for each row

-> begin.

-> insert into audit\_person (personid, oldname, newname, action) values (old.personid, old.name, new.name, 'updated');

-> end; //

mysql> .create trigger. trg\_person\_delete1

-> after delete on person

-> for each row

-> begin.

-> insert into logs (log\_message) values (concat('person deleted: ', old.name));

-> end; //

mysql> .create trigger. trg\_person\_delete2

-> after delete on person

-> for each row

-> begin.

-> insert into audit\_person (personid, action) values (old.personid, 'deleted');

-> end; //

mysql> delimiter;

mysql> delimiter//

mysql> create trigger trg\_employment\_insert1

-> after insert on employment

-> for each row

-> begin

-> insert into logs (log\_message) values (concat('new employment entry added for: ', new.personid));

-> end; //

mysql> create trigger trg\_employment\_insert2

-> after insert on employment

-> for each row

-> begin

->.insert into audit\_employment (employmentid, personid, action) values (new.employmentid, new.personid, 'inserted');

-> end;//

mysql> create trigger trg\_employment\_update1

-> after update on employment

-> for each row

-> begin

->.insert into logs (log\_message) values (concat('employment updated: personid ', old.personid));

-> end;//

mysql> .create trigger. trg\_employment\_update2

-> after update on employment

-> for each row

-> begin.

->.insert into audit\_employment (employmentid, oldbonus, newbonus, action) values (old.employmentid, old.bonus, new.bonus, 'updated');

-> end; //

mysql> .create trigger. trg\_employment\_delete1

-> after delete on employment

-> for each row

-> begin.

->,insert into logs (log\_message) values (concat('employment entry deleted for: ', old.personid));

-> end; //

mysql> create trigger trg\_employment\_delete2

-> after delete on employment

-> for. each row

-> begin.,

->.insert into audit\_employment (employmentid, personid, action) values (old.employmentid, old.personid, 'deleted');

-> end; //

mysql> delimiter ;

mysql> delimiter //

mysql> create trigger trg\_position\_insert1

-> after insert on position

-> for each row

-> begin.

-> . insert into logs (log\_message) values (concat('new position added: ', new.title));

mysql> create trigger trg\_position\_insert2

-> after insert on position

-> for each row

-> begin.

-> insert into audit\_position (positionid, action) values (new.positionid, 'inserted');

-> end; //

mysql> create trigger trg\_position\_update1

-> after update on position

-> for each row

-> begin.

-> .insert into logs (log\_message) values (concat('position updated: ', old.title, ' to ', new.title));

-> end;//

query ok, 0 rows affected (0.02 sec)

mysql> create trigger trg\_position\_update2

-> after update on position

-> for each row

-> begin

-> insert into audit\_position (positionid, oldtitle, newtitle, action) values (old.positionid, old.title, new.title, 'updated');

-> end; //

mysql>

mysql> create trigger trg\_position\_delete1

-> after delete on position

-> for each row

-> begin

-> insert into logs (log\_message) values (concat('position deleted: ', old.title));

-> end; //

mysql> create trigger trg\_position\_delete2

-> after delete on position

-> for each row

-> begin.

-> insert into audit\_position (positionid, action) values (old.positionid, 'deleted');

-> end;//

mysql>mysql> delimiter//

mysql> create. trigger. trg\_employmentposition\_insert1

-> after insert on employmentposition

-> for each row

-> begin.

-> .insert into logs (log\_message) values (concat('new employment position entry added for employmentid: ', new.employmentid));

-> end; //

mysql> create trigger trg\_employmentposition\_insert2

-> after insert on employmentposition

-> for each row

-> begin.

->.insert into audit\_employmentposition (employmentpositionid, action) values (new.employmentpositionid, 'inserted');

-> end;//

mysql> create trigger trg\_employmentposition\_update1

-> after update on employmentposition

-> for each row

-> begin.

-> insert into logs (log\_message) values (concat('employment position updated for employmentid: ', old.employmentid));

-> end;//

mysql> create trigger trg\_employmentposition\_update2

-> after update on employmentposition

-> for each row

-> begin.

->.insert into audit\_employmentposition (employmentpositionid, oldemploymentid, newemploymentid, action)

-> values (old.employmentpositionid, old.employmentid, new.employmentid, 'updated');

-> end; //

mysql> .create trigger. trg\_employmentposition\_delete1 after delete on employmentposition

-> for each row

-> begin.

-> insert into logs (log\_message) values (concat('employment position entry deleted for employmentid: ', old.employmentid));

-> end;//

mysql> .create trigger. trg\_employmentposition\_delete2 after delete on employmentposition

-> for each row

-> begin.

-> insert into audit\_employmentposition (employmentpositionid, action) values (old.employmentpositionid, 'deleted');

-> end;//

mysql> delimiter;

**creating fuctions and procedures**

mysql>mysql> delimiter//

mysql> create function getfullname(personid int)

-> returns var(255)

-> deterministic

-> reads sql data begin.

-> declare fullname var(255);

-> select concat(firstname, ' ', lastname) into fullname from person where personid = personid;

-> return fullname; end;//

mysql> create function countemployees(organizationid int) returns int

-> deterministic

-> reads sql data

-> begin.

-> declare employeecount int;

-> select count(\*) into employeecount

-> from employment

-> where organizationid = organizationid; -- ensure the organizationid column is correct

-> return employeecount; end; //

mysql>

mysql> delimiter;

mysql> delimiter//

mysql> create procedure addnewperson(

-> in firstname varchar(50),

-> in lastname varchar(50))

-> begin

-> insert into person (firstname, lastname) values (firstname, lastname);

-> insert into logs (log\_message) values (concat('new person added: ', firstname, ' ', lastname));

-> end; //

mysql> delimiter;

mysql> delimiter//

mysql> create. procedure updatepersonposition(

-> in personid int,

-> in newpositionid int)

-> begin

-> update employment

-> set positionid = newpositionid

-> where personid = personid;

-> insert into logs (log\_message) values (concat('updated position for personid: ', personid, ' to positionid: ', newpositionid));

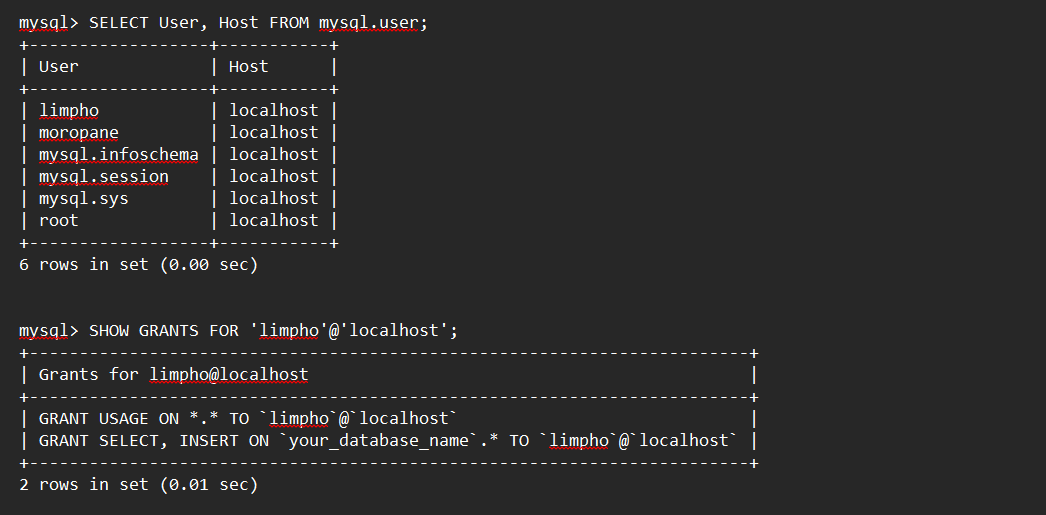
-> end; //

**cteating users and grants**

mysql> create user 'moropane'@'localhost' identified by 'moropane123';

mysql> grant select, insert on your\_database\_name.\* to 'limpho'@'localhost';

mysql> grant select, insert on your\_database\_name.\* to 'moropane'@'localhost';



*FIGURE 13. shows users available and grants*